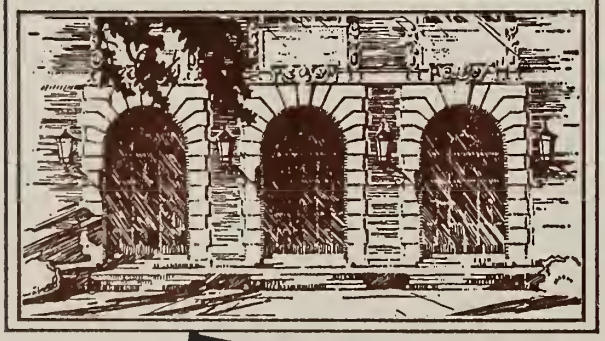

UIUCDCS-R-73-575

COO-1469-0225

DOCUMENTATION FOR DFASUB--A Program for the Solution of Simultaneous Implicit Differential and Nonlinear Equations

by

R. L. Brown
C. W. Gear

July 1973

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-73-575

DOCUMENTATION FOR DFASUB--A Program for the Solution of Simultaneous Implicit Differential and Nonlinear Equations*

by

R. L. Brown
C. W. Gear

July 1973

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

* Supported in part by the Atomic Energy Commission under contract US AEC AT(11-1)1469.

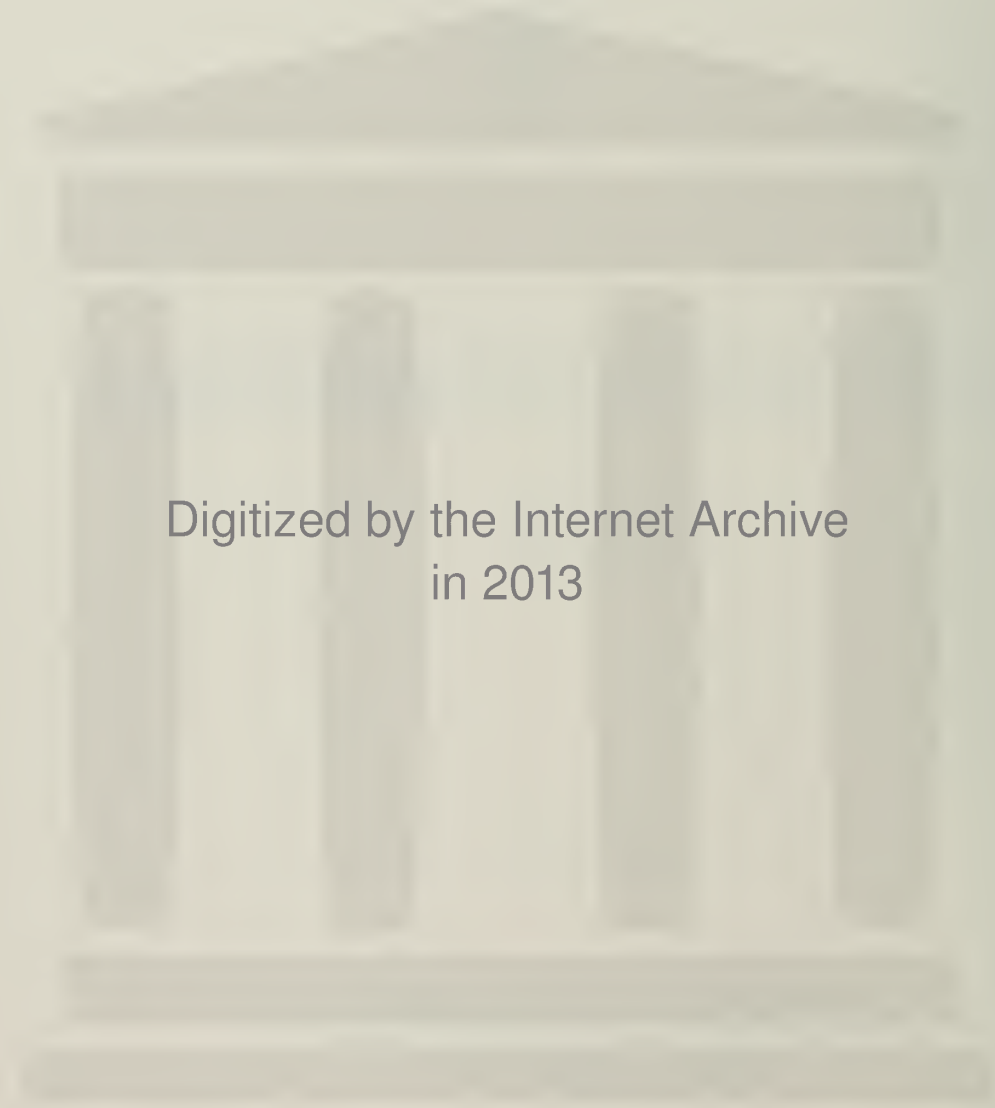

TABLE OF CONTENTS

1. INTRODUCTION

The purpose of the FORTRAN program DFASUB is to integrate a system of equations--described by a mixture of ordinary differential equations, nonlinear equations, and linear equations--using a discrete variable method. The equations are written in the form

$$f(\underline{y}, \underline{y}', t) + P\underline{v} = 0 \qquad (1.1)$$

where the vectors $\underline{y}$ and $\underline{y}' = d\underline{y}/dt$ are of length p, $\underline{v}$ is of length q-p, P is a q x (q-p) matrix, t is the independent variable, and $\underline{f}$ is a vector function of length q.

DFASUB uses the variable values to integrate the system using a multistep method designed for use with stiff ordinary differential equations which also works well with non-stiff equations and which will handle nonlinear equations. This paper deals with both the theory behind the integration method and the program itself. Section 2 presents the theory.

At present, the routine calls a number of other subroutines which are compiled by a general-purpose system [1]. The function of these will be described so they can be replaced by user-supplied FORTRAN subroutines. The form of the internal variables of the program is described in Section 3 to allow the user to write subroutines equivalent to those above as described in that section. Section 4 details calling parameters to allow a user to write a complete integration package built around DFASUB. Section 5 describes the detailed program logic of DFASUB.

This introduction, together with Sections 3 and 4, serve as a user's guide to DFASUB. Reading Section 2 is sufficient to understand the theory behind the routine. Someone interested in the programming aspects of the routine may read Sections 1, 3-5 to understand how DFASUB works.

2. THEORETICAL BACKGROUND

Suppose that the values of $\underline{y}$ are known (approximately) at a number of equally spaced points t_{n-i}, $i = 1,2,\ldots k$, where $t_{i+1} > t_i$. The backward differentiation formula gives the relation

$$h\underline{y}'_n = -\frac{1}{\beta_0}\left(\alpha_0\underline{y}_n + \alpha_1\underline{y}_{n-1} + \ldots + \alpha_k\underline{y}_{n-k}\right) \tag{2.1}$$

where $h = t_{i+1} - t_i > 0$. (The coefficients α_i and β_0 can be found in Gear [2], p. 217.) If this is substituted in (1.1) for $t = t_n$ we get

$$F_n(\underline{y}_n, \underline{v}_n) = \underline{f}(\underline{y}_n, -\frac{\alpha_0}{h\beta_0}\underline{y}_n + \Sigma_n, t_n) + P_n\underline{v}_n = 0 \tag{2.2}$$

where

$$\Sigma_n = -\frac{1}{h\beta_0}\left(\alpha_1\underline{y}_{n-1} + \ldots + \alpha_k\underline{y}_{n-k}\right)$$

is known. Hence, (2.2) is a system of q equations in the q unknowns $\underline{y}_n$ and $\underline{v}_n$. In general, they are nonlinear. If these are solved by a Newton-like iteration, we get

$$\begin{bmatrix}\underline{y}_{n,(m+1)}\\ \underline{v}_{n,(m+1)}\end{bmatrix} = \begin{bmatrix}\underline{y}_{n,(m)}\\ \underline{v}_{n,(m)}\end{bmatrix} - J^{-1}\,F_n(\underline{y}_{n,(m)}, \underline{v}_{n,(m)}) \tag{2.3}$$

where

$$J = \frac{\partial F_n}{\partial(\underline{y},\underline{v})} = \left[\begin{array}{c:c}\dfrac{\partial \underline{f}}{\partial \underline{y}} - \dfrac{\alpha_0}{h\beta_0}\dfrac{\partial \underline{f}}{\partial \underline{y}'} & P_n\end{array}\right] \tag{2.4}$$

and $\underline{y}_{n,(m)}$ and $\underline{v}_{n,(m)}$ are the q iterates for the solution $\underline{y}_n$ and $\underline{v}_n$ of (2.2). If accurate initial guesses $\underline{y}_{n,(0)}$ and $\underline{v}_{n,(0)}$ are used, very few iterations of (2.3) are necessary. In fact, accuracy is not needed in $\underline{v}_{n,(0)}$ as we show below that the iterates $\underline{y}_{n,(1)}$ and $\underline{v}_{n,(1)}$ are independent of $\underline{v}_{n,(0)}$ if round-off errors are ignored. To see this, compare the first iterate

$\hat{y}_{n,(1)}$ and $\hat{v}_{n,(1)}$ starting with $y_{n,(0)}$ and $\hat{v}_{n,(0)}$ with the first iterates $y_{n,(1)}$ and $v_{n,(1)}$ starting with $y_{n,(0)}$ and $v_{n,(0)}$. From (2.3)

$$\underline{z} = \begin{bmatrix} \hat{y}_{n,(1)} - y_{n,(1)} \\ \hat{v}_{n,(1)} - v_{n,(1)} \end{bmatrix} = \begin{bmatrix} 0 \\ \hat{v}_{n,(0)} - v_{n,(0)} \end{bmatrix} - J^{-1} P_n(\hat{v}_{n,(0)} - v_{n,(0)})$$

Hence,

$$J\underline{z} = J \begin{bmatrix} 0 \\ \hat{v}_{n,(0)} - v_{n,(0)} \end{bmatrix} - P_n(\hat{v}_{n,(0)} v_{n,(0)})$$

$$= \left[\begin{array}{c|c} \frac{\partial \underline{f}}{\partial \underline{y}} - \frac{\alpha_0}{h\beta_0} \frac{\partial \underline{f}}{\partial \underline{y}'} & P_n - P_n \end{array}\right] \begin{bmatrix} 0 \\ \hat{v}_{n,(0)} - v_{n,(0)} \end{bmatrix}$$

$$= 0$$

Since we assume J is non-singular (or the Newton iteration will not work), we see that $\underline{z} = 0$; hence, $\hat{y}_{n,(1)} = y_{n,(1)}$ and $\hat{v}_{n,(1)} = v_{n,(1)}$ are independent of $v_{n,(0)}$. Note that the only condition on J for this to be true is that the right-hand p columns of J be exactly P_n. The left-hand q-p columns can contain anything, so we need only approximate J in those positions.

Initial Guess for $y_{n,(0)}$

Since past values are known, a good initial guess can be found using polynomial extrapolation. We assume that we know y_{n-i}, $i \leq i \leq k$, and y'_{n-1}. (The latter was found at the last step or could be evaluated from (2.1) with n-1 replacing n.) With those values we can use a Hermite interpolation formula in the form

$$\underline{y}_{n,(0)} = h\bar{\beta}_1 \underline{y}'_{n-1} + \bar{\alpha}_1 \underline{y}_{n-1} + \ldots + \bar{\alpha}_k \underline{y}_{n-k} \qquad (2.5)$$

Since equal intervals are used, the $\bar{\alpha}_i$ and $\bar{\beta}_1$ are independent of n and h.

The Basic Algorithm

The initial guess is calculated from (2.5)--it is called the predicted value. $\underline{y}_{n,(0)}$ is set to $\underline{y}_{n-1}$. Equation (2.3) is iterated until $\underline{y}_{n,(m)}$ and $\underline{y}_{n,(m)}$ appear to have converged. (If they do not, the step size h is reduced as discussed later and the process is repeated.)

The error introduced by one step of (2.1) is known to be of the form

$$\frac{h^{k+1} y^{(k+1)}}{k+1}(\xi).$$

If the error in the numerical solution is a sufficiently smooth function of h and t, $h^{k+1} y^{(k+1)}(\xi_1) + O(h^{k+2})$ can be computed using a difference formula where ξ_1 is not necessarily the same point as ξ, but both ξ_1 and ξ are in the interval (t_{n-k}, t_n). This enables the error to be estimated under the assumption that $y^{(k+1)}$ changes slowly over the interval.

If the error estimate is too large, the result is rejected, and a smaller step is tried. Otherwise, the step is accepted and a suitable step size and order (k) is chosen for the next step.

Internal Representation of the Algorithm

We have used a Nordsieck [3] vector to represent past values of the data because of the ease of order changing and error estimation. We will describe this vector below for a single variable y_n and its associated past values $y_{n-1}, \ldots y_{n-k}$. This allows us to use the vector notation

$$\underline{y}_n = [y_n, hy'_n, y_{n-1}, \ldots y_{n-k+1}]^T$$

for the set of saved information concerning y at t_n. (T is the transpose operator.) When we step from t_{n-1} to t_n, we use (2.5) to predict $y_{n,(0)}$ from $\underline{y}_{n-1}$. At the same time, we save the values of $y_{n-1}, \ldots y_{n-k+1}$ from $\underline{y}_{n-1}$. We can represent this process by

$$\begin{bmatrix} y_{n,(0)} \\ 0 \\ y_{n-1} \\ y_{n-2} \\ \cdots \\ y_{n-k+1} \end{bmatrix} = \begin{bmatrix} \bar{\alpha}_1 & \bar{\beta}_1 & \bar{\alpha}_2 & \cdots & \bar{\alpha}_{k+1} & \bar{\alpha}_k \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ & \cdots & & & \cdots & \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \underline{y}_{n-1} \tag{2.6}$$

Next we iterate (2.3). To do this we must substitute $y_{n,(m)}$ and $\underline{v}_{n,(m)}$ into $F_n(y_n, v_n)$. Thus, from (2.2) we must evaluate

$$F_n(y_{n,(m)}, v_{n,(m)}) = f(y_{n,(m)}, -\frac{\alpha_0}{h\beta_0} y_{n,(m)} + \Sigma_n, t_n) + P_n v_{n,(m)} \tag{2.7}$$

We will normalize (2.1) so that $\alpha_0 = -1$. Let us write

$$\begin{aligned} hy'_{n,(m)} &= -\frac{\alpha_0}{\beta_0} y_{n,(m)} + h\Sigma_n \\ &= hy'_{n,(m-1)} + \frac{1}{\beta_0}(y_{n,(m)} - y_{n,(m-1)}) \quad \text{if } m > 0 \\ &= +\frac{1}{\beta_0}[\bar{\alpha}_1 y_{n-1} + \bar{\alpha}_2 y_{n-2} + \ldots + \bar{\alpha}_k y_{n-k} + h\bar{\beta}_1 hy'_{n-1}] \\ &\quad -\frac{1}{\beta_0}[\alpha_1 y_{n-1} + \ldots + \alpha_k y_{n-k}] \quad \text{if } m = 0 \end{aligned} \tag{2.8}$$

Hence,

$$hy'_{n,(0)} = \frac{\bar{\alpha}_1-\alpha_1}{\beta_0} y_{n-1} +\ldots+ \frac{\bar{\alpha}_k-\alpha_k}{\beta_0} y_{n-k} + \frac{\bar{\beta}_1}{\beta_0} hy'_{n-1}$$

$$\triangleq \gamma_1 y_{n-1} +\ldots+ \gamma_k y_{n-k} + \delta_1 hy'_{n-1} \tag{2.9}$$

Thus, (2.7) can be written as

$$F_n(y_{n,(m)}, v_{n,(m)}) = f(y_{n,(m)}, hy'_{n,(m)}, t_n) + P_n v_{n,(m)} \tag{2.10}$$

If we write $\underline{y}_{n,(m)} = [y_{n,(m)}, hy'_{n,(m)}, y_{n-1},\ldots, y_{n-k+1}]^T$, we see that we can combine (2.9) with (2.6) to get

$$\underline{y}_{n,(0)} = \begin{bmatrix} \bar{\alpha}_1 & \bar{\beta}_1 & \bar{\alpha}_2 & \cdots & \bar{\alpha}_{k-1} & \bar{\alpha}_k \\ \gamma_1 & \delta_1 & \gamma_2 & \cdots & \gamma_{k-1} & \gamma_k \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ & \cdots & & & \cdots & \\ 0 & 0 & 0 & & 1 & 0 \end{bmatrix} \underline{y}_{n-1} \triangleq B\underline{y}_{n-1} \tag{2.11}$$

This is called the predicted value of $\underline{y}_n$. Now we note from (2.8) that

$$\underline{y}_{n,(m+1)} = \underline{y}_{n,(m)} + \underline{c}(y_{n,(m+1)} - y_{n,(m)}) \tag{2.12}$$

where $\underline{c} = [1, \frac{1}{\beta_0}, 0,\ldots, 0]^T$ and $y_{n,(m+1)} - y_{n,(m)}$ is given by a component of (2.3).

The final step is to perform the transformation to a Nordsieck vector. The approximations y_n, y'_n, $y_{n-1},\ldots y_{n-k+1}$ determine a unique k-th degree polynomial. Let its scaled derivatives at t_n be y_n, hy'_n,... $h^k y_n^{(k)}/k!$. There is a unique non-singular k+1 by k+1 matrix Q such that if

$$\underline{z}_n = [y_n, hy_n^{(1)}, y^2 y_n^{(2)}/2,\ldots, h^k y_n^{(k)}/k!]^T$$

then

$$\underline{z}_n = Q\underline{y}_n .$$

With this we restate (2.11) as

$$\underline{z}_{n,0} = Q\underline{y}_{n,(0)}$$

$$= QBQ^{-1}\,\underline{z}_{n-1} \triangleq A\underline{z}_{n-1} \tag{2.13}$$

and (2.12) as

$$\underline{z}_{n,(m+1)} = Q\underline{y}_{n,(m+1)}$$

$$= \underline{z}_{n,(m)} + \underline{\ell}\,(\underline{y}_{n,(m+1)} - \underline{y}_{n,(m)}) \tag{2.14}$$

where

$$\underline{\ell} = Q\underline{c}$$

and

$$A = QBQ^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & k \\ & 0 & 1 & 3 & 6 & \vdots \\ & & 0 & 1 & 4 & \vdots \\ & & & 0 & 1 & \iddots \\ & & & & 0 & k \\ & & & & 0 & 1 \end{bmatrix}$$

The fact that A is a Pascal triangle matrix follows from the fact that the predictor performs a k-th order exact approximation $\underline{z}_{n,(0)}$ to $\underline{z}_n$. The advantage of this form is that the step size can be changed easily by multiplying the vector $\underline{z}_n$ by $C(\alpha) = \mathrm{diag}[1, \alpha, \alpha^2, \ldots, \alpha^k]$ where $\alpha = h_{new}/h_{old}$. The order can be reduced by dropping a column from A, a

row from A and $\underline{z}_n$, and changing $\underline{\ell}$. The order can be increased by adding a row and column to A, changing $\underline{\ell}$, and estimating $h^{k+1}y^{(k+1)}/(k+1)!$. This is estimated by the last element of $(\underline{z}_n - \underline{z}_{n-1})/(k+1)$. (This is also used in computing the error estimate.) The values of $\underline{\ell}$ can be found in Gear [2], p. 217.

The choice of step size and order depends on several things. First, if the error estimate after the corrector converges is too large, the step size is reduced to h/4 and the step is taken over; if this fails to work after three such attempts, the order is reduced to 1 and a final attempt is made before giving up.

If the convergent corrector value is within error bounds, then every k steps at order k the step size at the original order, order k-1, and k+1 is calculated with a bias toward the lowest order (with the least work required) to get the largest step size. This is only done every k steps due to stability considerations [4].

The error estimator

$$L_h(\underline{y}_n) = \sum_{j=0}^{k} \alpha_j \, y_{n-j} + h\beta_j \, y'_{n-j}$$

is the local truncation error for the method in use if $\alpha_0 = -1$. If y(t) has a continuous (k+1)-th derivative, then

$$L_h(y(t)) = C_{k+1} \, y^{(k+1)}(\xi) \, h^{k+1}$$

if the method is of order k and $t-h \leq \xi \leq t$. We can define

$$C_{k+1} = \frac{L_h(t^{k+1})}{h^{k+1}(k+1)!}$$

since for a k-th order method applied to the (k+1)-th order polynomial t^{k+1}, the error term $L_h(t^{k+1})$ is known and

$$(t^{k+1})^{(k+1)} = (k+1)!$$

3. REPRESENTATION OF VARIABLES; SUBROUTINES

DFASUB requires several variable arrays for integrating a system; these arrays are described below. The remaining arguments in the calling sequence are briefly described following those. For notational convenience N is the number of equations q, NY is the number of nonlinear equations p, and NL = N-NY. The notation A(B)C means "from A to C in steps of B".

Y(J,I) for I = 1(1)NY and J = 1(1)7 contains the current value of the differential and nonlinear variables in its first row. Each subsequent row J contains the (J-1)-th derivative of the variable y_i multiplied by H**(J-1)/(J-1)!. Thus, Y contains all the elements necessary for a Taylor series expansion without requiring the derivatives to be multiplied by $H^k/k!$ before use. Thus, the predictor step consists of multiplying Y(I,J) by the Pascal triangle matrix as described in Section 2.

YL(I) for I = 1(1)NL holds the value of the linear variables, corresponding to $\underline{v}$ in Section 2.

DY(I) holds the result of calling the evaluation subroutine DIFFUN(T,G,DY,Y,YL,HINV); for I = 1(1)NY it contains the difference $y_i' - f(y_i)$ for y_i = Y(1,I), y_i' = Y(2,I)*HINV where HINV is the inverse of the step size H. For I = NY+1(1)N the error in the linear equations stored in YL(I) is in DY(I). Thus, DY(I) is essentially the correction to the values in Y(1,I) and YL(J) before the Newton iteration.

SAVE(J,I) for J = 1(1)7, I = 1(1)NY holds the initial values of the nonlinear variables at the beginning of an integration step. If the step fails, the independent variable T(1) is restored to its original values and Y(J,I) takes on the values in SAVE(J,I). YLSV(I) for I = 1(1)NL serves the same purpose for the linear variables in YL(I).

ERROR(I) for I = 1(1)NY stores the sum of the corrections computed during each integration step. It is initialized to zero after the predictor evaluation and is incremented by F1(I)--the improved corrector--after each corrector step. If the integration step is successful, all elements of Y(J,I) are updated by Y(J,I) = Y(J,I) + A(J)*ERROR(I). The difference for ERROR(I) between two integration steps is used to calculate the next higher derivative element when the best step size for order k+1 is being determined. When it is known that on the next step a new value of H will be calculated, ERROR(I) is saved in ERSV(I) for I = 1(1)NY.

YMAX(I) for I = 1(1)NY stores the maximum of 1.0 or the largest value of $|Y(1,I)|$ computed. YMAX(I) is used in the error evaluation for an absolute error test when Y(1,I) has never exceeded 1.0 and a relative error test when it has.

A(J), J = 1(1)NQ+1, stores the updating factors for the higher order derivative terms in Y. It corresponds to the vector $\underline{\ell}$ in Section 2.

G(J) is storage for global variables, parameters that may be easily changed from one integration to the next. They allow a user to experiment with different parameters in attempts to optimize given systems under simulation.

T(J) contains variables that are dependent only on G(*) and the independent variable T(1). Thus, they need to be evaluated only for each value of the independent variable.

PW(I,J) is the matrix J.

The remaining calling arguments to DFASUB follow:

EPS is the error per step criterion that must be met by the L2 norm of the error.

EQN and VAR are two arrays that are required by routine MATSET (see following) when MATSET is compiled by the general-purpose system [1]. When this is not the case, they can be dummy arrays.

HMAX is the largest step size that DFASUB will take. It should be chosen less than the period of any known periodic solution. Otherwise, no special care need be taken in choosing it.

HMIN is the minimum step size taken. H is the initial step size used unless it causes convergence problems, in which case a new step size is chosen.

JSTART is a start indicator. When it is zero on a call to DFASUB, the routine initializes itself. When it is 1, it assumes that initialization has taken place and continues integration. On exit, it contains the current order.

KFLAG is the output completion code. Its meanings are

+1 successful integration

-1 error test failure for H > HMIN

-2 too many floating point errors

-3 the corrector failed to converge for H > HMIN

-4 the corrector failed for even the first order method

It is possible to write into the main program a routine to try possible remedies when DFASUB quits with an unsuccessful completion code making use of KFLAG and the computed GO TO in FORTRAN. After these remedial measures are taken, JSTART should be set to 1 and DFASUB called again.

M is the dimension of DY, normally the same as N.

MAXDER is the maximum order method used, never more than 6.

M1 is the number of variables in Y(1,J), J = 1(1)M1, which are included in the error test. If the error in any group of variables is not critical, then they should be placed at the end of the Y array and M1 set to exclude them in the error test. N is the total number of equations. NL is the number of linear equations.

TEND is the final value of T(1). Integration stops when this value is reached.

Subroutines

The dependent variables in DFASUB fall into three classes: those dependent only on global terms or parameters in the G(*) array, variables described by linear equations which have Jacobian elements dependent at most on the arrays G(*) and T(*) so these are the elements of the matrix P of Section 2, and variables whose entries to the Jacobian matrix change whenever some other variable changes its value. To handle these differences efficiently, five subroutines called by DFASUB are concerned with computing the Jacobian matrix and performing the corrector iteration (2.3).

Since the elements in the Jacobian have different dependencies, they can be recomputed at different times by

MATSET (A(2),DY,EPS,EQN,G,HINV,O,M,MF,N,NY,IT,PW,F1,T,VAR,Y,YL).

When IT = 1, only those values in the Jacobian that are fixed for one set of parameters G(*) are calculated. This need be done only once per integration. When IT = 2, only those values that are dependent on T(*) and G(*) are computed. This is done whenever a new step is

started, causing a new value $T(1)_{new} = T(1)_{old} + H$ to be computed, and whenever the step size is reduced and the integration step restarted. When IT = 3, all remaining elements are computed. This must be done at each corrector iteration.

The corrector iteration (2.3) does not actually multiply by J^{-1} but rather performs a symbolic inversion of the sparse matrix PW. Again, the symbolic inversion of PW is broken up according to the dependence of each element on T, G, and Y. MATIN1(PW) inverts just those elements that are dependent only on G. MATIN2(PW) inverts elements dependent on T and G. MATIN3 inverts all remaining elements.

MATMUL(PW,DY,F1) performs the equivalent of solving (PW)(F1) = DY for F1; DY holds the values of F(y, v) = F(y, y', t) + P(t)v; and F1 is a vector used in the correction step.

DIFFUN(T,G,DY,Y,YL,HINV) is the differentiating subroutine which places $y_i' - f_i(\underline{y})$ = HINV*Y(2,I) - f(y(1,I)) into DY(I) for I = 1(1)NY, and the correction to the linear equations into DY(I), I = NY+1(1)N.

The remaining subroutines are not connected with the differential algebraic equation system, but are used for convenience.

KNTSPI(1) is an assembly language function that keeps track of floating point overflow and underflow, and returns a value of three or more if enough such errors are encountered to invalidate the results of the computation.

COPYZ(S,X,L) copies the L values of single precision array X into single precision array S. The actual variables used are double precision arrays so L is twice the size of the actual arrays. If X = Y, S = SAVE, we have L = 14*NY. For X = YL, S = YLSV, we have L = 2*NY.

S2(T,G) evaluates those variables stored in T(*) that are dependent on T(1), the independent variable; and G(*), the global parameters.

4. USER PROGRAM FOR DFASUB

To write a program to use DFASUB all the subroutines called by it must be present in some form. The subroutine DIFFUN, which is supplied by a system compiler in the general-purpose system, must be provided to solve the system

$$f(y, y', t) + P(t)\underline{v} = 0.$$

The calling arguments are the T(*) and G(*) arrays, the Y array containing y_i in Y(1,I) and $H*y_i'$ in Y(2,I), HINV = 1/H, and v_i in the array YL. The system can be put into the proper form by expressing the equations as

$$\begin{aligned} 0 &= y_1' - (f_1(Y,YL,T,G) + P_1\underline{v}) \\ &\vdots \\ 0 &= y_{NY}' - (f_{NY}(Y,YL,T,G) + P_{NY}\underline{v}) \\ 0 &= f_{NY+1}(Y,YL,T,G) + P_{NY+1}\underline{v} \\ &\vdots \\ 0 &= f_N(Y,YL,T,G) + P_N\underline{v} \end{aligned}$$

for P_I the I-th row of P(t).

The zeros are then replaced by

$$DY(I),\ I = 1(1)N.$$

An example system is to be found in [5]. It is

$$0 = y_i' - s + (r-y_i)^2 + \sum_{i=1}^{4} b_{ij}\, y_j, \qquad i = 1(1)4$$

for

$$r = (y_1 + y_2 + y_3 + y_4)/2$$

$$s = \sum_{i=1}^{4} (r-y_i)^2/2$$

$$[b_{ij}] = B = \begin{bmatrix} 447.5+\varepsilon & -452.5+\varepsilon & -47.5+\varepsilon & -52.5-\varepsilon \\ -452.5+\varepsilon & 447.5+\varepsilon & 52.5+\varepsilon & 47.5-\varepsilon \\ -47.5+\varepsilon & 52.5+\varepsilon & 447.5+\varepsilon & 452.5+\varepsilon \\ -52.5-\varepsilon & 47.5-\varepsilon & 452.5-\varepsilon & 447.5+\varepsilon \end{bmatrix}$$

for $\varepsilon = .00025$.

$$0 = y_5' + y_1\, y_6' + y_1'\, y_6$$

$$0 = 2y_6 = y_6^3 - y_1 + v_1 - 1 - e^{-t}$$

$$= v_1 + F_6$$

$$0 = v_1 - v_2 + y_1\, y_6$$

$$= v_1 - v_2 + F_7$$

$$0 = v_1 + v_2 + 5y_1\, y_2$$

$$= v_1 + v_2 + F_8$$

for $y_5^{(0)} = y_6^{(0)} = 1$, $v_1^{(0)} = -2$, $v_2^{(0)} = -3$, $y_i^{(0)} = -1$, $i = 1(1)4$.

Appendix I contains the FORTRAN routine DIFFUN(T,G,DY,Y,YL,HINV) that is used to evaluate this system as well as a sample routine S2(T,G) to solve T(2) = EXP(-T(1)).

The computation of the Jacobian can be split up into the three subsets, each of which is evaluated by MATSET depending on the parameter IT. Three different inversions are also required. If the user would rather not write these four routines, a simpler routine MATSET can be written so that nothing is done when IT is equal to 1 or 2, and the entire Jacobian matrix is generated when MATSET (...,3,...) is called. The evaluation of the Jacobian can be done by a special routine that computes $\frac{\partial f_i}{\partial y_j}$ and $\frac{\partial f_i}{\partial y'_j}$ according to an analytic formula, or else numerical differencing as in [6] requiring N calls to DIFFUN may be used. MATIN3 can be replaced by a call to a standard Gaussian elimination subroutine, while MATIN1 and MATIN2 are made dummy subroutines. MATMUL can be replaced by a back substitution subroutine written to work with the routine in MATIN3.

Two implementations of this procedure are possible. The easiest one is to write a subroutine which has a place in its calling sequence for each of the parameters in the calling sequence in DFASUB, and then manipulate these parameters, possibly preparing them for a call to a second subroutine. This has been done in the remainder of Appendix I. For example, MATSET has dummy variables for EQN and VAR, two arrays MATSET as compiled by the general-purpose system use but are not needed in this version, which performs numerical differencing to compute PW when IT is 3 and does nothing when IT is not 3. MATIN3(PW) simply calls Moler's routine [7] DECOMP. Since DECOMP requires the number of variables N, this has been provided in the unnamed COMMON block in MATMUL and DFASUB. MATMUL calls Moler's routine SOLVE which also needs N. Both of these routines communicate through an array IP, which is provided in a named COMMON/IPP/ and dimensioned at least as large as the system.

The second implementation requires that the user remove all superfluous subroutine calls and change the calling sequence of the routines he does use, thus saving the overhead of dummy subroutine calls and extraneous preparation of data. This has the disadvantage that if DFASUB is updated, all this work must be repeated on the new version; whereas if the subroutine calls are left alone, those that work with one version of DFASUB will probably work with any new version.

Usable versions of COPYZ and KNTSPI are also included in Appendix I. If the computer in use has some way to count suppressed underflow and overflow under program control, KNTSPI can be programmed to return a value greater than 3 when too many errors occur.

Finally, DFASUB must provide the initial values of Y(1,I), Y(2,I), and YL(J). The general-purpose system calls a program DIFMF3 [8] which, if given either Y(1,I) or Y(2,I) for each I and YL(J), will find all necessary initial values. If all initial values Y(1,I) are known, then by setting Y(2,I) = 0 for I = 1,NY, calling DIFFUN once and setting

$$Y(2,I) = -DY(I),$$

good approximations can be given to all necessary initial values.

5. PROGRAM LOGIC

An anthropomorphic view of the actions of DIFSUB while solving a system of equations follows. A main program specifically written for the PDP-8/GRAPHICS system [1] provides the calling sequence that enters the subroutine DIFSUB and the subroutines which DIFSUB calls are compiled by the system loaded with DIFSUB. A detailed description of what each of these do is to be found in Section 3.

The program is entered with JSTART = 0. The bookkeeping variables that record the number of steps, NS, and the number of matrix evaluations, NW, are initialized to 0. Both y and y' are expected to be in array Y(1,I) and Y(2,I). This can be accomplished by a call to the steady state problem package DIFMF3 [8]. Y(2,I) is scaled by H, YMAX(I) is set to 1.0, the order NQ is set to 1, and the variables ℓ_j are placed in A(J), J = 1(1)NQ+1. Various error test criteria are computed in E, EUP, and EDWN; ENQ1, ENQ2, and ENQ3 held 1/(2*NQ), 1/(2*(NQ+1), 1/(2*(NQ+2) the power to which the ratio of computed error to desired error must be taken to find the ratio of new to old step size.

MATSET is called with IT = 1 to evaluate any constant elements of PW; MATIN1 inverts these elements.

The value of Y(J,I), J = 1(1)NQ+1, I = 1(1)NY are saved in SAVE(J,I) in case they are needed to restart the program; YL(I), I = 1(1)NL is saved in YLSV(I). H, T, and NQ are also saved. DIFSUB then calls S2 which evaluates any variables there are functions only of T and G. MATSET is called with an argument of 2 to evaluate and update any part of the Jacobian matrix PW which would change as a result of the call to S2. DIFSUB then calls MATIN2 which does an inversion of parts of PW that change as a result of calling S2 and MATSET (...,2,...).

DIFSUB multiplies Y(J,I) by the Pascal triangle matrix as described in [6]. This replaces y_{i-1} with $y_{i,(0)}$, the predicted value of the nonlinear variables.

The corrector step is a long loop that starts by initializing ERROR to 0. DIFSUB calls DIFFUN(T,G,DY,Y,YL,HINV) which places in DY the updated $y'_{i,(1)}$ value. MATSET (...,3,...) is now called to update the Jacobian as a result of having the predicted value of y_i and the corrected value of y'_i available. MATIN3(PW) is called to make any changes in the LU decomposition as a result of the call to MATSET (...,3,...). IWEVAL is set to -1 so that all of this re-evaluation of PW need not be done again unless it seems necessary. MATMUL(PW,DY,F1) is called which does the back substitution that computes the amount by which Y(J,I) and YL(I) are to be changed to hold the corrected value. YL(I), Y(1,I) and Y(2,I) are corrected by A(J)*F1(I); $||\Delta y_{(1)}||_2$ is then computed. If $||\Delta y||_2$ is less than BND, the computed error bound for the order, then the program continues. If not, MATMUL is again called, YL(I), Y(1,I), Y(2,I) are corrected, $\Delta y_{(2)}$ is added to $\Delta y_{(1)}$ and placed in ERROR(*).

Actually, not only the present corrector term but also an estimate of the next corrector term is compared to BND. Since the L2 norm of the i-th corrector term $||\text{ERROR}^i||_2$ is approximately $R*||\text{ERROR}^{i-1}||_2$ for R constant throughout each small region of integration, the predicted next value of the corrector can also be compared to BND and the method is considered to have converged if

$$\text{MIN}(||\text{ERROR}^i||, R*||\text{ERROR}^{i+1}||*2) < \text{BND}.$$

R is initially set to $||\text{ERROR}^2||/||\text{ERROR}^1||$ and updated at each step.

Assume that the corrector term did converge in two corrections. Then the estimated error

$$||\text{ERROR(I)/YMAX(I)}||_2$$

is computed; and if this is less than E, the maximum error bound allowed to have the real error per step less than EPS, the program continues. Otherwise, a better value of H for this or one lower order is computed as described below, KFLAG is reduced by 2, the original values for this step are returned, and the step is repeated for the new step size. Should KFLAG reach -5, indicating three steps taken with three different step sizes yet none gave an acceptably small error, the order is reduced to 1 and another attempt is made. If this is unsuccessful, KFLAG is set to -4 and DIFSUB makes an error return to its calling program. If at some time H becomes less than HMIN, KFLAG is set to -1 and DIFSUB makes an error return.

Otherwise, the rest of the Y(J,I) array, if NQ > 1, is updated using the contents of ERROR(I) by Y(J,I) = Y(J,I) + ERROR(I)*A(J), then DIFSUB returns to the beginning of the program. JSTART has been set to 1 so no initialization occurs. The integration step procedes as in the first step, except that after NQ successful steps ERROR(I) is stored in ERSV(I) and the next step (if successful) uses this and other information to check for a better step size and order.

Since we also have an estimate for y_n and $y_n^{(k)}/k!$ stored in the array Y(7,I), and since ERROR(I) stores the current step's accumulated corrections for Y(1,I), the backward difference of the last component of Y(K,N) is ERROR(I)*A(K) which is an estimate for

$$\frac{h^{k+1}\, y_n^{(k+1)}}{k!} .$$

Also, since ERROR(I) corresponding to the last y_{n-1} are saved in ERSV(I), we know that

$$\text{ERSV(I)} - \text{ERROR(I)} \doteq h^{(k+2)}\, y^{(k+2)}(t_n)/k!.$$

These values can be used to compute the best step size and order in the following ways.

After the predictor-corrector method converges, we can check to see if the truncation error was within bounds by seeing if

$$\text{EPS}^2 \geq \sum_{I=1}^{M1} \left(\frac{C_{k+1}\, h^{k+1}\, y_I^{(k+1)}}{\text{YMAX(I)}}\right)^2$$

$$= \sum_{I=1}^{M1} \left(\frac{C_{k+1}\text{*ERROR(I)*K!*A(K)}}{\text{YMAX(I)}}\right)^2$$

by comparing

$$E = (\text{EPS}/C_{k+1}\ \text{A(K)*K!})^2$$

to

$$D = \sum_{I=1}^{M1} \left(\frac{\text{ERROR(I)}}{\text{YMAX(I)}}\right)^2.$$

When we want to see about changing order and step size (after K+N*9, N = 0,1,..., successful steps at order K and step size H), we can compute the best step size to use at the current order, order 1 higher and order 1 lower. Since

$$\frac{D}{E} \doteq \left(\frac{H_{NEW}}{H}\right)^2 (p+1)$$

by computing $\text{PR2} = 1.2(D/E)^{1/2(k+1)}$,

$$\text{HNEW} = \text{H/PR2}$$

will give the best step size for order K (1.2 is a heuristic constant to compensate for ignoring the $O(h^{k+2})$ terms).

If one order lower method is used, then

$$EPS^2 \leq \Sigma\left(\frac{C_k\, h^k\, y_I^{(k)}}{YMAX(I)}\right)^2 = \Sigma\left(\frac{C_k * y(k,I) * p!}{YMAX(I)}\right)^2$$

letting

$$EDWN = (EPS/C_k * k!)^2$$

and

$$\hat{D} = \Sigma\left(\frac{Y(K,I)}{YMAX(I)}\right)^2$$

then

$$PR1 = 1.3(\hat{D}/EDWN)^{1/2p}$$

gives the factor for the best step size for order k-1 methods.

Finally, if order one higher is used, we need

$$EPS^2 \leq \Sigma\left(\frac{C_{k+2}\, h^k\, y_I^{(k)}}{YMAX(I)}\right)^2$$

$$= \Sigma\left(\frac{C_{k+2}\ (ERSV(I)-ERROR(I))A(K)*K!)}{YMAX(I)}\right.$$

Letting $EUP = (EPS/C_{k+2} * A(K) * p!)^2$ and

$$D = \Sigma\left(\frac{D/ERROR(I)}{YMAX(I)}\right)^2$$

then

$$\frac{D}{E} = \left(\frac{HNEW}{H}\right)^{2\ (p+2)}$$

and

$$PR3 = 1.4\left(\frac{D}{EUP}\right)^{1/2\ (p+2)}$$

gives the best step size. The 1.3 and 1.4 terms bias the method toward not changing the order or picking one order lower, respectively, since these would minimize overhead calculations.

Once a new step size has been computed, the array Y(J,I) must be changed to reflect the new step size since $Y(J,I) = Y^{(J-1)}h^{(J-1)}/(J-1)!$. This is accomplished by multiplying the elements of SAVE(J,I) (if the single step error were too large and the step is being repeated) or of Y(J,I) (if a new step size is being prepared for the next step) by $(H/HOLD)^{(J-1)}$.

This is done in a DO loop after statement 750 or 800 depending on whether SAVE or Y is supplying the values to be reached. If no significant improvement in step size can be made, IDOUB is set to 9 (or other large integer) and the step size is re-evaluated again if the steps continue to be successful.

If, contrary to our assumption, the corrector steps do not converge after two tries, then H is reduced to H/4 unless IWEVAL = 0 indicating that PW should be re-evaluated first. Y(J,I) is scaled to reflect this new step size and a new integration step is attempted. If H cannot be reduced, KFLAG is set to -3 and DIFSUB makes an error return.

Each time a new step is entered, DIFSUB checks for T > TEND and KFLAG < 0. If either of these is true, the program exits to the calling routine with JSTART set to the present value of the order NQ, HNEW set to the best H for the next step, H set to the present H, and Y(J,I) set to its last successful value.

The entry point REDSUB is used to re-enter the program after problems with the matrix processing routines.

This section, while describing the general program logic, leaves out many details of programming that would be unwieldly to describe here. The theory and some ideas about the programming techniques are in the other sections, and after reading these and using the program copy in Appendix II, this section should be sufficient to understand the program logic.

LIST OF REFERENCES

[1] Gear, C. W., "An Interactive Graphic Modeling System," Department of Computer Science Report No. 318, University of Illinois at Urbana-Champaign, 1969.

[2] Gear, C. W., NUMERICAL INITIAL VALUE PROBLEMS IN ORDINARY DIFFERENTIAL EQUATIONS, Prentice-Hall: New Jersey, 1971.

[3] Nordsieck, A., "On the Numerical Integration of Ordinary Differential Equations," Math. Comp., 16, 1962, pp. 22-49.

[4] Gear, C. W. and Watanabe, D. S., "Stability and Convergence of Variable Order Multistep Methods," submitted to SIAM Journal on Numerical Analysis.

[5] Gear, C. W., "Simultaneous Numerical Solution of Differential-Algebraic Equations," IEEE Trans. CT, 18, #1, 1971, pp. 89-95.

[6] Gear, C. W., "DIFSUB for the Solution of Ordinary Differential Equations," CACM, 14, #3, 1972, pp. 185-190.

[7] Moler, C. B., "Matrix Computations with FORTRAN and Paging," CACM, 15, #4, 1972, pp. 268-270.

[8] van Melle, B., "An Operating Manual for DIFMF3," Department of Computer Science File No. 870, University of Illinois at Urbana-Champaign, 1972.

APPENDIX I

SAMPLE PROGRAM

```
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(7,6),YL(2),SAVE(7,8),YLSV(2),PW(84),
     1 G(16),T(2),DY(8),ERSV(6),ERROR(6),F1(8),EQN(1),VAR(1),YMAX(8)
      DATA G/447.50025,-452.49975,-47.49975,-52.50025,
     + -452.49975,447.50025,52.50025,47.49975,-47.49975,
     + 52.50025,447.50025,452.49975,-52.50025,47.49975,
     + 452.49975,447.50025/
C*************************************************
C*    SUPPRESS OVERFLOW AND UNDERFLOW
C*************************************************
      DATA N,NL,M,EPS,HMAX,HMIN,H,T,TEND/8,2,8,1.D-4,
     + 5.D2,1.D-10,1.D-4,0.D0,1.D0,1.D3/
      CALL ERRSET(208,256,-1,1)
      CALL ERRSET(207,256,-1,1)
C*************************************************
C*    SET INITIAL VALUES AND ZERO FIRST DERIVATIVES.
C*************************************************
      DO 10 I=1,4
   10 Y(1,I)=-1.
      Y(1,5)=1.
      Y(1,6)=1.
      DO 20 I=1,6
   20 Y(2,I)=0.D0
      YL(1)=-2.
      YL(2)=-3.
      M=N
C*************************************************
C*    FIND FIRST DERIVATIVES OF NONLINEAR VARIABLES
C*************************************************
      CALL DIFFUN(T,G,DY,Y,YL,1.)
      DO 30 I=1,6
   30 Y(2,I)=-DY(I)
      JSTART=0
   35 CALL DFASU8(DY,EPS,EQN,ERROR,ERSV,F1,G,H,HMAX,HMIN,N,
     1 JSTART,KFLAG,M,6,6,N,NL,PW,SAVE,T,TEND,VAR,Y,YL,
     2  YLSV,YMAX)
      WRITE(6,999)KFLAG
  999 FORMAT('      KFLAG =',I6)
   50 STOP
      END



      SUBROUTINE S2(T,G)
      REAL*8 T(2)
      T(2)=DEXP(-T(1))
      RETURN
      END



      SUBROUTINE DIFFUN(T,G,DY,Y,YL,HINV)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION G(4,4),DY(8),Y(7,6),YL(2),T(2)
      R=(Y(1,1)+Y(1,2)+Y(1,3)+Y(1,4))/2.




      S=0.
      DO 20 I=1,4
   20 S=S+(R-Y(1,I))**2/2.
      DO 30 I=1,4
      DY(I)=H(NV*Y(2,I)-S+(R-Y(1,I))**2
      DO 25 J=1,4
   25 DY(I)=DY(I)+G(I,J)*Y(1,J)
   30 CONTINUE
      DY(5)=H(NV*(Y(2,5)+Y(1,1)*Y(2,6)+Y(2,1)*Y(1,6))
      DY(6)=2.*Y(1,6)+Y(1,6)**3-Y(1,1)+YL(1)-1.-T(2)
      DY(7)=YL(1)-YL(2)+Y(1,1)*Y(1,6)
      DY(8)=YL(1)+YL(2)+5.*Y(1,1)*Y(1,2)
      RETURN
      END
```

```
      SUBROUTINE COPYZ(S,Y,L)
      DIMENSION Y(1),S(1)
      DO 10 J=1,L
   10 S(J)=Y(J)
      RETURN
      END
```

```
      FUNCTION KNTSPI(I)
      KNTSPI=I
      RETURN
      END
```

```
      SUBROUTINE MATIN1(P)
      RETURN
      END
```

```
      SUBROUTINE MATIN2(P)
      RETURN
      END
```

```
      SUBROUTINE MATSET(A2,DY,EPS,D4,G,HINV,D5,D6,D7,N,NY,ICHK,
     1  PW,F1,T,D9,Y,YL)
      IMPLICIT REAL*8(A-H,Q-Z)
      DIMENSION DY(1),G(1),PW(1),T(1),Y(7,1),YL(1),F1(1)
C*    SEE IF THIS IS A CALL TO BE FULLY HANDLED.
      IF(ICHK.NE.3) GO TO 100
      NL=N-NY
      NN=N*N
      DO 20 I=1,NN
   20 PW(I)=0.
      DO 40 J=1,NY
C*    SAVE COLUMN ELEMENTS
      F=Y(1,J)
      E=Y(2,J)
      R=EPS*DMAX1(EPS,DABS(F),DABS(G))
C*    FIND A DIFFERENCE ELEMENT = MAX(EPS**2,EPS*Y(1,J),EPS*Y(2,J))
      Y(1,J)=Y(1,J)+R
      Y(2,J)=Y(2,J)-A2*R
      CALL DIFFUN(T,G,F1,Y,YL,HINV)
C*                    DF   A(2) DF
C*    ASSIGN VALUES TO -- - ---- --
C*                    DY     H  DY'
      DO 30 I=1,N
   30 PW(I+(J-1)*N)=(F1(I)-DY(I))/R
C*    RESTORE COLUMN ELEMENTS
      Y(2,J)=E
   40 Y(1,J)=F
C*    IF ANY LINEAR ELEMENTS, FIND DF/DYL.
      IF(NL.EQ.0)GO TO 100
      DO 80 J=1,NL
      F=YL(J)
      R=EPS*DMAX1(EPS,DABS(F))
      YL(J)=YL(J)+R
      CALL DIFFUN(T,G,F1,Y,YL,HINV)
      DO 70 I=1,N
   70 PW(I+(J+NY-1)*N)=(F1(I)-DY(I))/R
   80 YL(J)=F
  100 RETURN
      END
```

```
      SUBROUTINE MATIN3(PW)
      IMPLICIT REAL*8 (A-H,Q-Z)
      DIMENSION PW(1)
      COMMON N
      COMMON /IPP/ IP(40)
      CALL DECOMP(N,N,PW,IP)
      RETURN
      END
```

```
      SUBROUTINE MATMUL(PW,DY,F1)
      IMPLICIT REAL*8 (A-H,Q-Z)
      COMMON N
      COMMON /IPP/ IP(40)
      DIMENSION PW(1),DY(1),F1(1)
      DO 10 I=1,N
   10 F1(I)=DY(I)
      CALL SOLVE(N,N,PW,F1,IP)
      RETURN
      END
```

```
      SUBROUTINE DECOMP(N,NDIM,A,IP)
      IMPLICIT REAL*8(B-H,O-Z)
C
C     MATRIX TRIANGULARIZATION BY GAUSSIAN ELIMINATION.
C
C   INPUT...
C     N = ORDER OF MATRIX
C     NDIM = DECLARED DIMENSION OF ARRAY A.
C     A = MATRIX TO BE TRIANGULARIZED. (FOR STIFF METHODS, A IS SINGLE
C                 PRECISION;  ALL OTHER VARIABLE ARE DOUBLE PRECISION.)
C   OUTPUT...
C     A(I,J), I.LE.J  = UPPER TRIANGULAR FACTOR, U.
C     A(I,J),   I.GT.J   = MULTIPLIERS  = LOWER TRIANGULAR FACTOR, I-L.
C     IP(K),  K.LT.N = INDEX OF K-TH PIVOT ROW.
C     IP(N)  = (-1)**(NUMBER OF INTERCHANGES) OR 0.
C   USE 'SOLVE' TO OBTAIN SOLUTION OF LINEAR SYSTEM.
C   DETERM(A) = IP(N)*A(1,1)*A(2,2)*...*A(N,N).
C   IF IP(N) = 0, A IS SINGULAR, SOLVE WILL DIVIDE BY ZERO.
C
      DIMENSION A(NDIM,NDIM),IP(NDIM)
      DIMENSION B(4,4)
      COMMON NFNS,NW,B
      NW = NW + 1
      IP(N) = 1
      DO 6  K=1,N
      IF(K.EQ.N) GO TO 5
      KP1 = K+1
      M = K
      DO 1  I=KP1,N
      IF(ABS(A(I,K)).GT.ABS(A(M,K)))   M=I
    1 CONTINUE
      IP(K) = M
      IF(M.NE.K)  IP(N) = -IP(N)
      T = A(M,K)
      A(M,K) = A(K,K)
      A(K,K) = T
      IF(T.EQ.0) GO TO 5
      DO 2  I=KP1,N
    2    A(I,K) = -A(I,K) / T
      DO 4  J=KP1,N
      T = A(M,J)
      A(M,J) = A(K,J)
      A(K,J) = T
      IF(T.EQ.0.) GO TO 4
      DO 3   I=KP1,N
    3     A(I,J) = A(I,J) + A(I,K)*T
    4  CONTINUE
    5 IF(A(K,K).EQ.0.) IP(N) = 0
    6 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SOLVE(N,NDIM,A,B,IP)
      IMPLICIT REAL*8 (B-H,O-Z)
C
C   SOLUTION OF LINEAR SYSTEM, A*X = B.
C
C   INPUT...
C     N = ORDER OF MATRIX.
C     NDIM = DECLARED DIMENSION OF ARRAY A.
C     A = TRIANGULARIZED MATRIX OBTAINED FROM 'DECOMP'.
C     B = RIGHT HAND SIDE VECTOR.
C     IP = PIVOT VECTOR OBTAINED FROM 'DECOMP'.
C   OUTPUT...
C     B = SOLUTION VECTOR, X.
      DIMENSION A(NDIM,NDIM), B(NDIM), IP(NDIM)
      IF(N.EQ.1) GO TO 9
      NM1 = N-1
      DO 7  K=1,NM1
         KP1 = K + 1
         M = IP(K)
       T=B(M)
      B(M) = B(K)
      B(K) = T
      DO 7  I=KP1,N
    7 B(I) = B(I) + A(I,K)*T
      DO 8   KB=1,NM1
      KM1 = N - KB
      K = KM1 + 1
      B(K) = B(K)/A(K,K)
      T = -B(K)
      DO 8  I=1,KM1
    8 B(I) = B(I) + A(I,K)*T
    9 B(1) = B(1)/A(1,1)
      RETURN
      END
```

APPENDIX II

DFASUB LISTING

```
      SUBROUTINE DFASUB (DY,EPS,EQN,ERROR,ERSV,                          DFAS 001
     +   F1,G,H,HMAX,HMIN,                                               DFAS 002
     +   JSTART,KFLAG,M,MAXDER,M1,N,                                     DFAS 003
     +   NL,PW,SAVE,T,TEND,VAR,Y,                                        DFAS 004
     +   YL,YLSV,YMAX)                                                   DFAS 005
      IMPLICIT REAL*8 (A-H,O-Z)                                          DFAS 006
C*********************************************************************DFAS 007
C*                                                                     DFAS 008
C*    THE PARAMETERS TO THE SUBROUTINE DIFSUB HAVE                     DFAS 009
C*    THE FOLLOWING MEANINGS:                                          DFAS 010
C*                                                                     DFAS 011
C*      N        THE NUMBER OF VARIABLES.                              DFAS 012
C*      NL       THE NUMBER OF LINEAR VARIABLES                        DFAS 013
C*     (NY = N-NL  IS THE NUMBER OF VARIABLES WITH DERIVATIVES)        DFAS 014
C*      M1       THE NUMBER OF EQUATIONS TO TAKE PART IN THE ERROR TEST. DFAS 015
C*      TEND     END CRITERION                                         DFAS 016
C*      T        THE INDEPENDENT VARIABLE.                             DFAS 017
C*      G        AN ARRAY OF GLOBAL VARIABLES                          DFAS 018
C*      Y        A 7 BY NY ARRAY CONTAINING THE DEPENDENT VARIABLES    DFAS 019
C*                 AND THEIR SCALED DERIVATIVES. Y(J+1,I) CONTAINS     DFAS 020
C*                 THE J-TH DERIVATIVE OF Y(I) SCALED BY               DFAS 021
C*                 H**J/FACTORIAL(J), H THE CURRENT STEP SIZE.         DFAS 022
C*                 ON THE FIRST ENTRY, THE CALLER SUPPLIES             DFAS 023
C*                 Y(1,I) AND Y(2,I), UNSCALED. (IF THE CALL TO        DFAS 024
C*                 DIFSUB WAS PRECEDED BY A CALL TO DIFMF3, THE        DFAS 025
C*                 USER NEED NOT TOUCH Y AT ALL). THE PROGRAM          DFAS 026
C*                 WILL SCALE Y(2,I) BY H. ON ANY SUBSEQUENT           DFAS 027
C*                 ENTRY, THE PROGRAM ASSUMES THAT THE Y VALUES        DFAS 028
C*                 HAVE NOT BEEN CHANGED SINCE THE LAST EXIT           DFAS 029
C*                 FROM DIFSUB, AND WILL SCALE THESE VALUES IF         DFAS 030
C*                 THE CALLER HAS CHANGED H.                           DFAS 031
C*                   IF IT IS DESIRED TO INTERPOLATE TO NON-MESH       DFAS 032
C*                 POINTS THESE VALUES CAN BE USED. IF THE CURRENT     DFAS 033
C*                 STEP SIZE IS H AND THE VALUE AT T+E IS NEEDED,      DFAS 034
C*                 FORM S = E/H AND THE COMPUTE                        DFAS 035
C*                                  NQ                                 DFAS 036
C*                   Y(I)(T+E) =   SUM  Y(J+1,I)*S**J                  DFAS 037
C*                                 J=0                                 DFAS 038
C*      YL       AN ARRAY OF NL VARIABLES THAT APPEAR ONLY LINEARLY.   DFAS 039
C*                 CALLER MUST SUPPLY VALUES FOR THESE VARIABLES.      DFAS 040
C*      SAVE     AN ARRAY OF LENGTH AT LEAST 7*N.                      DFAS 041
C*      H        THE STEP SIZE TO BE ATTEMPTED ON THE NEXT STEP.       DFAS 042
C*                 H MAY BE ADJUSTED UP OR DOWN BY THE PROGRAM        DFAS 043
C*                 IN ORDER TO ACHEIVE AN ECONOMICAL INTEGRATION.      DFAS 044
C*                 HOWEVER, IF THE H PROVIDED BY THE USER DOES         DFAS 045
C*                 NOT CAUSE A LARGER ERROR THAN REQUESTED, IT         DFAS 046
C*                 WILL BE USED.  TO SAVE COMPUTER TIME, THE USER IS   DFAS 047
C*                 ADVISED TO USE A FAIRLY SMALL STEP FOR THE FIRST    DFAS 048
C*                 CALL.  IT WILL BE AUTOMATICALLY INCREASED LATER.    DFAS 049
C*      HMIN     THE MINIMUM STEP SIZE THAT WILL BE USED FOR THE       DFAS 050
C*                 INTEGRATION.  NOTE THAT ON STARTING THIS MUST BE    DFAS 051
C*                 MUCH SMALLER THAN THE AVERAGE H EXPECTED SINCE      DFAS 052
C*                 A FIRST ORDER METHOD IS USED INITIALLY.             DFAS 053
C*      HMAX     THE MAXIMUM ALLOWABLE STEP SIZE                       DFAS 054
C*      EPS      THE ERROR TEST CONSTANT.  SINGLE STEP ERROR ESTIMATES DFAS 055
C*                 DIVIDED BY  YMAX(I)  MUST BE LESS THAN THIS         DFAS 056
C*                 IN THE EUCLIDEAN NORM. THE STEP AND/OR ORDER IS     DFAS 057
C*                 ADJUSTED TO ACHEIVE THIS.                           DFAS 058
C*      YMAX     A VECTOR OF LENGTH NY WHICH CONTAINS THE MAXIMUM      DFAS 059
C*                 OF EACH Y SEEN SO FAR. ON THE FIRST CALL, THESE     DFAS 060
C*                 WILL BE INITIALIZED AS YMAX(I) = MAX(1,|Y(1,I)|)    DFAS 061

C*      ERROR    A VECTOR OF LENGTH NY.                                DFAS 062
C*      KFLAG    A COMPLETION CODE WITH THE FOLLOWING MEANINGS:        DFAS 063
C*                   +1  THE INTEGRATION WAS SUCCESSFUL.               DFAS 064
C*                   -1  THE ERROR TEST FAILED FOR H > HMIN.           DFAS 065
C*                   -3  THE CORRECTOR FAILED TO CONVERGE FOR          DFAS 066
C*                         H > HMIN.                                   DFAS 067
C*                   -2  TOO MANY FLOATING-POINT EXCEPTIONS            DFAS 068
C*                         OCCURRED DURING LAST STEP.                  DFAS 069
C*                   -4  THE CORRECTOR FAILED THREE TIMES WITH         DFAS 070
C*                         EVEN THE FIRST-ORDER METHOD.                DFAS 071
```

```
C*       JSTART  AN INPUT INDICATOR WITH THE FOLLOWING MEANINGS:          DFAS 072
C*                    0   PERFORM THE FIRST STEP. THE ROUTINE            DFAS 073
C*                          INITIALIZES ITSELF, SCALES THE DERIV-        DFAS 074
C*                          ATIVES IN Y(2,I) AND THEN PERFORMS           DFAS 075
C*                          THE INTEGRATION UNTIL T > TEND.              DFAS 076
C*                          ANY SUBSEQUENT CALLS SHOULD BE MADE          DFAS 077
C*                          WITH JSTART = 1.                             DFAS 078
C*                    1   CONTINUE FROM THE LAST STEP, INTEGRATING       DFAS 079
C*                          UNTIL T > TEND.                              DFAS 080
C*                  JSTART IS SET TO NQ, THE CURRENT ORDER OF            DFAS 081
C*                  THE METHOD, AT EXIT.                                 DFAS 082
C*       MAXDER  THE MAXIMUM DERIVATIVE THAT SHOULD BE USED IN THE       DFAS 083
C*                 METHOD. IT MUST NOT EXCEED 6.                         DFAS 084
C*       PW    A VECTOR OF LENGTH N**2+20 (REAL*4),                      DFAS 085
C*                 GENERATED BY MATSET AND USED BY MATINV,MATMUL.        DFAS 086
C*       YLSV     A VECTOR OF LENGTH NL.                                 DFAS 087
C*       DY       A VECTOR OF LENGTH M, OUTPUT OF DIFFUN.                DFAS 088
C*       ERSV     A VECTOR OF LENGTH NY.                                 DFAS 089
C*       F1       A VECTOR OF LENGTH N, OUTPUT OF MATMUL.                DFAS 090
C*       EQN,VAR    VECTORS USED BY MATSET.                              DFAS 091
C*                                                                       DFAS 092
C***********************************************************************DFAS 093
      DIMENSION T(1),G(1),Y(7,1),YL(1),SAVE(7,1),YMAX(1)                 DFAS 094
      DIMENSION ERROR(1),PW(1),YLSV(1),DY(1),ERSV(1)                     DFAS 095
      DIMENSION F1(1),EQN(1),VAR(1),A(7),PERTST(6,3)                     DFAS 096
      DATA PERTST /4.0,9.0,16.0,25.0,36.0,49.0,9.0,16.0,                 DFAS 097
     +             25.0,36.0,49.0,64.0,1.0,1.0,0.25,                     DFAS 098
     +             2.7889E-2,1.70569E-3,6.83929E-5/                      DFAS 099
      DATA MF /2/, KZILCH /0/                                            DFAS 100
C*************************************************                       DFAS 101
C*    THIS COMMUNICATES THE VALUE OF N TO MATMUL AND MATIN3.             DFAS 102
C*************************************************                       DFAS 103
      COMMON N99                                                         DFAS 104
C                                                                        DFAS 105
   1  FORMAT (/I5,I4,D11.2,D10.2,6D15.5/(31X,6D15.5))                    DFAS 106
   2  FORMAT (31X,6D15.5)                                                DFAS 107
   3  FORMAT ('1MF =',I2,'   N =',I3,'   NL =',I3,'   EPS =',            DFAS 108
     +        D9.2,'   TEND =',D9.2,'   H =',D9.2//)                     DFAS 109
   4  FORMAT ('  NS  NW      H',8X,'T(1)',8X,                           DFAS 110
     +        'Y(1,*) AND YL(*)'//)                                      DFAS 111
C                                                                        DFAS 112
      NY = N-NL                                                          DFAS 113
C*    LENGTH IN WORDS OF Y, YL (FOR COPY) :                              DFAS 114
      LCOPYY = NY*14                                                     DFAS 115
      LCOPYL = NL*2                                                      DFAS 116
      N99=N                                                              DFAS 117
      IF (MAXDER .GT. 6) MAXDER = 6                                      DFAS 118
         HINV=2.5                                                        DFAS 119
         CALL DIFFUN (T,G,F1,Y,YL,HINV)                                  DFAS 120
         CALL MATSET (0,DY,EPS,EQN,G,HINV, 0,M,MF,N,NY,1,                DFAS 121
     +   PW,F1,T,VAR,Y,YL)                                               DFAS 122


         CALL MATIN1 (PW)                                                DFAS 123
      IF (JSTART .GT. 0) GO TO 60                                        DFAS 124
      WRITE (6,3) MF,N,NL,EPS,TEND,H                                     DFAS 125
      WRITE (6,4)                                                        DFAS 126
C*************************************************                       DFAS 127
C*    ON THE FIRST CALL SET THE ORDER TO ONE. INITIALIZE                 DFAS 128
C*    THE VECTOR YMAX, AND SCALE THE FIRST DERIVATIVES OF Y BY H.        DFAS 129
C*************************************************                       DFAS 130
      NS = 0                                                             DFAS 131
      NW = 0                                                             DFAS 132
      DO 20 J = 1,NY                                                     DFAS 133
        YMAX(J) = DMAX1(1.0D0,DABS(Y(1,J)))                              DFAS 134
  20    Y(2,J) = Y(2,J)*H                                                DFAS 135
      NQ = 1                                                             DFAS 136
      BR = 1.0                                                           DFAS 137
      ASSIGN 100 TO IRET                                                 DFAS 138
      GO TO 221                                                          DFAS 139
C*************************************************                       DFAS 140
C*    SET THE COEFFICIENTS THAT DETERMINE THE ORDER AND THE METHOD       DFAS 141
C*    TYPE.  E IS A TEST FOR ERRORS OF THE CURRENT ORDER NQ.             DFAS 142
C*    EUP IS TO TEST FOR INCREASING THE ORDER, EDWN FOR                  DFAS 143
C*    DECREASING THE ORDER.                                              DFAS 144
C*************************************************                       DFAS 145
```

```
170   CONTINUE                                                          DFAS 146
      GO TO (221,222,223,224,225,226),NQ                                DFAS 147
221   A(2) = -1.0D0                                                     DFAS 148
      GO TO 230                                                         DFAS 149
222   A(2) = -1.5D0                                                     DFAS 150
      A(3) = -0.5D0                                                     DFAS 151
      GO TO 230                                                         DFAS 152
223   A(2) = -1.83333333333333333                                       DFAS 153
      A(3) = -1.0D0                                                     DFAS 154
      A(4) = -0.16666666666666667                                       DFAS 155
      GO TO 230                                                         DFAS 156
224   A(2) = -2.08333333333333333                                       DFAS 157
      A(3) = -1.45833333333333333                                       DFAS 158
      A(4) = -0.4166666666666667                                        DFAS 159
      A(5) = -0.04166666666666667                                       DFAS 160
      GO TO 230                                                         DFAS 161
225   A(2) = -2.8333333333333333                                        DFAS 162
      A(3) = -1.875D0                                                   DFAS 163
      A(4) = -0.7083333333333333                                        DFAS 164
      A(5) = -0.125D0                                                   DFAS 165
      A(6) = -0.0083333333333333333                                     DFAS 166
      GO TO 230                                                         DFAS 167
226   A(2) = -2.45D0                                                    DFAS 168
      A(3) = -2.2555555555555555                                        DFAS 169
      A(4) = -1.2083333333333333                                        DFAS 170
      A(5) = -0.2430555555555555                                        DFAS 171
      A(6) = -0.02916666666666667                                       DFAS 172
      A(7) = -0.0013888888888888889                                     DFAS 173
230   K = NQ+1                                                          DFAS 174
      IDOUB = NQ                                                        DFAS 175
      ENQ1 = 0.5/FLOAT(NQ)                                              DFAS 176
      ENQ2 = 0.5/FLOAT(K)                                               DFAS 177
      ENQ3 = 0.5/FLOAT(NQ + 2)                                          DFAS 178
      PEPSH = EPS**2                                                    DFAS 179
      E    = PERTST(NQ,1)*PEPSH                                         DFAS 180
      EUP = PERTST(NQ,2)*PEPSH                                          DFAS 181
      EDWN= PERTST(NQ,3)*PEPSH                                          DFAS 182
      BND = (EPS*ENQ3)**2                                               DFAS 183


      IWEVAL = 1                                                        DFAS 184
      GO TO IRET, (100,250,630,751)                                     DFAS 185
C*****************************************************                 DFAS 186
C*    IF THE CALLER HAS CHANGED H, SCALE THE Y VARIABLES.              DFAS 187
C*    HNEW IS THE STEP SIZE USED ON THE LAST CALL.                     DFAS 188
C*****************************************************                 DFAS 189
  60  IF (H .EQ. HNEW) GO TO 100                                        DFAS 190
      R = H/HNEW                                                        DFAS 191
      ASSIGN 100 TO IRET                                                DFAS 192
      GO TO 800                                                         DFAS 193
C*****************************************************                 DFAS 194
C*    BEFORE EXIT, JSTART IS SET TO THE ORDER OF THE METHOD,           DFAS 195
C*    AND THE CURRENT VALUE OF H IS SAVED.                             DFAS 196
C*****************************************************                 DFAS 197
  70  KFLAG = -2                                                        DFAS 198
  80  JSTART = NQ                                                       DFAS 199
      HNEW = H                                                          DFAS 200
      RETURN                                                            DFAS 201
C*****************************************************                 DFAS 202
C*    PRINT DATA RELEVANT TO THE STEP, AND TAKE ANOTHER STEP           DFAS 203
C*    IF T < TEND.                                                     DFAS 204
C*****************************************************                 DFAS 205
  90  NS = NS+1                                                         DFAS 206
      WRITE (6,1) NS,NW,H,T(1),(Y(1,I),I=1,NY)                          DFAS 207
      IF (NL .GT. 0) WRITE (6,2) (YL(I),I=1,NL)                         DFAS 208
      CALL ANSWER(Y,F1,T)                                               DFAS 209
         IF (KNTSPI(0) .GT. 2) GO TO 70                                 DEAS 210
      IF (KFLAG .LT. 0) GO TO 80                                        DFAS 211
      IF (T(1) .GE. TEND) GO TO 80                                      DFAS 212
      JSTART = 1                                                        DFAS 213
C*****************************************************                 DFAS 214
C*    BEGIN BY SAVING INFORMATION FOR POSSIBLE RESTARTS.               DFAS 215
C*****************************************************                 DFAS 216
 100  CALL COPYZ (SAVE,Y,LCOPYY)                                        DFAS 217
      CALL COPYZ (YLSV,YL,LCOPYL)                                       DFAS 218
```

```
      RACUM = 1.0                                                       DFAS 219
      KFLAG = 1                                                         DFAS 220
      HOLD = H                                                          DFAS 221
      NQOLD = NQ                                                        DFAS 222
      TOLD = T(1)                                                       DFAS 223
      KZILCH = 1                                                        DFAS 224
C***********************************************************************DFAS 225
C*    THIS SECTION COMPUTES THE PREDICTED VALUES BY EFFECTIVELY        *DFAS 226
C*    MULTIPLYING THE SAVED INFORMATION BY THE PASCAL TRIANGLE         *DFAS 227
C*    MATRIX.                                                          *DFAS 228
C***********************************************************************DFAS 229
 250  T(1) = T(1)+H                                                     DFAS 230
      CALL S2(T,G)                                                      DFAS 231
         CALL MATSET (0,DY,EPS,EQN,G,HINV, 0,M,MF,N,NY,2,               DFAS 232
     +   PW,F1,T,VAR,Y,YL)                                              DFAS 233
         CALL MATIN2 (PW)                                               DFAS 234
      HINV = 1.D0/H                                                     DFAS 235
      DO 260 J = 2,K                                                    DFAS 236
        J3 = K+J-1                                                      DFAS 237
        DO 260 J1 = J,K                                                 DFAS 238
          J2 = J3-J1                                                    DFAS 239
          DO 260 I = 1,NY                                               DFAS 240
 260        Y(J2,I) = Y(J2,I) + Y(J2+1,I)                               DFAS 241
C***********************************************************************DFAS 242
C*      UP TO 3 CORRECTOR ITERATIONS ARE TAKEN.  CONVERGENCE IS TESTED *DFAS 243
C*      BY REQUIRING CHANGES TO BE LESS THAN BND WHICH IS DEPENDENT ON *DFAS 244


C*      THE ERROR TEST CONSTANT.                                       *DFAS 245
C*         THE SUM OF THE CORRECTIONS IS ACCUMULATED IN THE ARRAY      *DFAS 246
C*      ERROR(I).  IT IS EQUAL TO THE K-TH DERIVATIVE OF Y MULTIPLIED  *DFAS 247
C*      BY  H**K/(FACTORIAL(K-1)*A(K)), AND IS THEREFORE PROPORTIONAL  *DFAS 248
C*      TO THE ACTUAL ERRORS TO THE LOWEST POWER OF H PRESENT. (H**K)  *DFAS 249
C***********************************************************************DFAS 250
      DO 270 I = 1,NY                                                   DFAS 251
 270    ERROR(I) = 0.0                                                  DFAS 252
      DO 430 L = 1,3                                                    DFAS 253
        CALL DIFFUN (T,G,DY,Y,YL,HINV)                                  DFAS 254
C***********************************************************************DFAS 255
C*      IF THERE HAS BEEN A CHANGE OF ORDER OR THERE HAS BEEN TROUBLE  *DFAS 256
C*      WITH CONVERGENCE, PW IS RE-EVALUATED PRIOR TO STARTING THE      DFAS 257
C*      CORRECTOR ITERATION IN THE CASE OF STIFF METHODS.  IWEVAL IS   *DFAS 258
C*      THEN SET TO -1 AS AN INDICATOR THAT IT HAS BEEN DONE.          *DFAS 259
C***********************************************************************DFAS 260
        IF (IWEVAL .LT. 1 .AND. L .GT. 1) GO TO 280                     DFAS 261
         CALL MATSET (A(2),DY,EPS,EQN,G,HINV, 0,M,MF,N,NY,3,            DFAS 262
     +   PW,F1,T,VAR,Y,YL)                                              DFAS 263
          CALL MATIN3 (PW)                                              DFAS 264
          KFLAG = 1                                                     DFAS 265
          IWEVAL = -1                                                   DFAS 266
          NW = NW+1                                                     DFAS 267
280      CALL MATMUL (PW,DY,F1)                                         DFAS 268
        IF (NL .LE. 0) GO TO 300                                        DFAS 269
        DO 290 I = 1,NL                                                 DFAS 270
 290      YL(I) = YL(I) - F1(I+NY)                                      DFAS 271
 300    CONTINUE                                                        DFAS 272
      DEL = 0.0D0                                                       DFAS 273
        DO 420 I = 1,NY                                                 DFAS 274
          Y(1,I) = Y(1,I) - F1(I)                                       DFAS 275
          Y(2,I) = Y(2,I) + A(2)*F1(I)                                  DFAS 276
          ERROR(I) = ERROR(I) + F1(I)                                   DFAS 277
      DEL = DEL + (F1(I)/YMAX(I))**2                                    DFAS 278
 420      CONTINUE                                                      DFAS 279
      IF(L.GE.2) BR = DMAX1(.9*BR,DEL/DEL1)                             DFAS 280
      DEL1 = DEL                                                        DFAS 281
      IF(DMIN1(DEL,BR*DEL*2.0).LE.BND) GO TO 490                        DFAS 282
 430    CONTINUE                                                        DFAS 283
C***********************************************************************DFAS 284
C*    THE CORRECTOR ITERATION FAILED TO CONVERGE IN 3 TRIES.  VARIOUS  *DFAS 285
C*    POSSIBILITIES ARE CHECKED FOR.  IF H IS ALREADY HMIN AND         *DFAS 286
C*    THIS IS EITHER ADAMS METHOD OR THE STIFF METHOD IN WHICH THE     *DFAS 287
C*    MATRIX PW HAS ALREADY BEEN RE-EVALUATED, A NO CONVERGENCE EXIT   *DFAS 288
C*    IS TAKEN. OTHERWISE THE MATRIX PW IS RE-EVALUATED AND/OR THE     *DFAS 289
C*    STEP IS REDUCED TO TRY AND GET CONVERGENCE.                      *DFAS 290
C***********************************************************************DFAS 291
```

```
 440  T(1) = TOLD                                                    DEAS 292
      IF (IWEVAL) 445,455,450                                        DEAS 293
 445  IF (H .LE. HMIN*1.0000001) GO TO 460                           DEAS 294
 450  RACUM = RACUM*0.25                                             DEAS 295
 455  CONTINUE                                                       DEAS 296
      GO TO 750                                                      DEAS 297
 460  KFLAG = -3                                                     DEAS 298
 470  CALL COPYZ (Y,SAVF,LCOPYY)                                     DEAS 299
      CALL COPYZ (YL,YLSV,LCOPYL)                                    DEAS 300
      H = HOLD                                                       DEAS 301
      NQ = NQOLD                                                     DEAS 302
      GO TO 90                                                       DEAS 303
C*************************************************                  DEAS 304
C*    THE CORRECTOR CONVERGED AND NOW THE ERROR TEST IS MADE.        DEAS 305


C*************************************************                  DEAS 306
 490  D = 0.0                                                        DEAS 307
      DO 500 I = 1,M1                                                DEAS 308
 500    D = D + (ERROR(I)/YMAX(I))**2                                DEAS 309
      IWEVAL = 0                                                     DEAS 310
      IF (D .GT. E) GO TO 540                                        DEAS 311
C*************************************************                  DEAS 312
C*    THE ERROR TEST IS OKAY, SO THE STEP IS ACCEPTED. IF IDOUB      DEAS 313
C*    NOW BECOMES NEGATIVE, A TEST IS MADE TO SEE IF THE STEP        DEAS 314
C*    SIZE CAN BE INCREASED AT THIS ORDER OR ONE HIGHER OR           DEAS 315
C*    LOWER. THE CHANGE IS MADE ONLY IF THE STEP CAN BE IN-          DEAS 316
C*    CREASED BY AT LEAST 10%. IDOUB IS SET TO NQ TO PREVENT         DEAS 317
C*    FURTHER TESTING FOR A WHILE. IF NO CHANGE IS MADE, IDOUB       DEAS 318
C*    IS SET TO 9.                                                   DEAS 319
C*************************************************                  DEAS 320
      IF (K .LT. 3) GO TO 520                                        DEAS 321
      DO 510 J = 3,K                                                 DEAS 322
        DO 510 I = 1,NY                                              DEAS 323
 510      Y(J,I) = Y(J,I) + A(J)*ERROR(I)                            DEAS 324
 520  KFLAG = +1                                                     DEAS 325
      IDOUB = IDOUB-1                                                DEAS 326
      IF (IDOUB) 550,525,700                                         DEAS 327
 525  DO 530 I = 1,M1                                               DEAS 328
 530    ERSV(I) = ERROR(I)                                           DEAS 329
      GO TO 700                                                      DEAS 330
C*************************************************                  DEAS 331
C*    THE ERROR TEST FAILED. IF JSTART = 0, THE DERIVATIVES          DEAS 332
C*    IN THE SAVE ARRAY ARE UPDATED. TESTS ARE THEN MADE TO          DEAS 333
C*    FIX THE STEP SIZE AND PERHAPS REDUCE THE ORDER. AFTER          DEAS 334
C*    RESTORING AND SCALING THE Y VARIABLES, THE STEP IS             DEAS 335
C*    RETRIED.                                                       DEAS 336
C*************************************************                  DEAS 337
 540  IF (JSTART .GT. 0) GO TO 548                                   DEAS 338
      DO 544 I = 1,NY                                                DEAS 339
 544    SAVE(2,I) = Y(2,I)                                           DEAS 340
 548  KFLAG = KFLAG - 2                                              DEAS 341
      IF (H .LE. HMIN) GO TO 740                                     DEAS 342
      T(1) = TOLD                                                    DEAS 343
      IF (KFLAG .LE. -5) GO TO 720                                   DEAS 344
 550  PR2 = (D/E)**ENQ2*1.2                                          DEAS 345
      L = 0                                                          DEAS 346
      IF (NQ .LE. 1) GO TO 570                                       DEAS 347
        D = 0                                                        DEAS 348
        DO 560 J = 1,M1                                              DEAS 349
 560      D = D+(Y(K,J)/YMAX(J))**2                                  DEAS 350
        PR1 = (D/EDWN)**ENQ1*1.3                                     DEAS 351
        IF (PR1 .GE. PR2) GO TO 570                                  DEAS 352
        PR2 = PR1                                                    DEAS 353
        L = -1                                                       DEAS 354
 570  IF (KFLAG .LT. 0  .OR.  NQ .GE. MAXDER) GO TO 590              DEAS 355
        D = 0                                                        DEAS 356
        DO 580 J = 1,M1                                              DEAS 357
 580      D = D+((ERROR(J)-ERSV(J))/YMAX(J))**2                      DEAS 358
        PR1 = (D/EUP)**ENQ3*1.4                                      DEAS 359
        IF (PR1 .GE. PR2) GO TO 590                                  DEAS 360
        PR2 = PR1                                                    DEAS 361
        L = 1                                                        DEAS 362
 590  R = 1.0/AMAX1(PR2,1.E-5)                                       DEAS 363
      IF (KFLAG .LT. 0  .OR.  R .GE. 1.1) GO TO 600                  DEAS 364
        IDOUB = 9                                                    DEAS 365
        GO TO 700                                                    DEAS 366
```

```
 600  NEWQ = NQ+L                                                    DFAS 367
      K = NEWQ+1                                                     DFAS 368
      IF (NEWQ .LE. NQ) GO TO 620                                    DFAS 369
      R1 = A(NEWQ)/FLOAT(NEWQ)                                       DFAS 370
      DO 610 J = 1,NY                                                DFAS 371
 610    Y(K,J) = ERROR(J)*R1                                         DFAS 372
 620  CONTINUE                                                       DFAS 373
C*****************************************************             DFAS 374
C*    IF THE STEP WAS OKAY, SCALE THE Y VARIABLES IN ACCORDANCE      DFAS 375
C*    WITH THE NEW VALUE OF H. IF KFLAG < 0, HOWEVER, USE THE        DFAS 376
C*    SAVED VALUES (IN SAVE AND YLSV). IN EITHER CASE, IF THE        DFAS 377
C*    ORDER HAS CHANGED IT IS NECESSARY TO FIX CERTAIN PARAMETERS    DFAS 378
C*    BY CALLING THE PROGRAM SEGMENT AT LINE 170.                    DFAS 379
C*****************************************************             DFAS 380
      IDOUB = NQ                                                     DFAS 381
      IF (NEWQ .EQ. NQ) GO TO 630                                    DFAS 382
      NQ = NEWQ                                                      DFAS 383
      ASSIGN 630 TO IRET                                             DFAS 384
      GO TO 170                                                      DFAS 385
 630  IF (KFLAG .GT. 0) GO TO 670                                    DFAS 386
      RACUM = RACUM*R                                                DFAS 387
      GO TO 750                                                      DFAS 388
 670  R = DMAX1(DMIN1(HMAX/H,R),HMIN/H)                              DFAS 389
      H = H*R                                                        DFAS 390
      IWEVAL = 1                                                     DFAS 391
      ASSIGN 700 TO IRET                                             DFAS 392
      GO TO 800                                                      DFAS 393
 700  DO 710 I = 1,M1                                                DFAS 394
 710    YMAX(I) = DMAX1(YMAX(I),DABS(Y(1,I)))                        DFAS 395
      GO TO 90                                                       DFAS 396
C*****************************************************             DFAS 397
C*    THE ERROR TEST HAS NOW FAILED THREE TIMES, SO THE              DFAS 398
C*    DERIVATIVES ARE IN BAD SHAPE. RETURN TO FIRST-ORDER            DFAS 399
C*    METHOD AND TRY AGAIN. OF COURSE, IF NQ = 1 ALREADY,            DFAS 400
C*    THEN THERE IS NO HOPE AND WE EXIT WITH KFLAG = -4.             DFAS 401
C*****************************************************             DFAS 402
 720  IF (NQ .EQ. 1) GO TO 735                                       DFAS 403
      NQ = 1                                                         DFAS 404
      IDOUB = 1                                                      DFAS 405
      ASSIGN 751 TO IRET                                             DFAS 406
      GO TO 170                                                      DFAS 407
 735  NQOLD = 1                                                      DFAS 408
      KFLAG = -4                                                     DFAS 409
      GO TO 470                                                      DFAS 410
 740  KFLAG = -1                                                     DFAS 411
      GO TO 90                                                       DFAS 412
C*****************************************************             DFAS 413
C*    THIS SECTION RESTORES THE SAVED VALUES OF Y AND YL, SCALING    DFAS 414
C*    THE Y DERIVATIVES AS NECESSARY, AND THEN RETURNS TO THE        DFAS 415
C*    PREDICTER LOOP.                                                DFAS 416
C*****************************************************             DFAS 417
 750  H = HOLD*RACUM                                                 DFAS 418
      H = DMAX1(HMIN,DMIN1(H,HMAX))                                  DFAS 419
 751  RACUM = H/HOLD                                                 DFAS 420
      R1 = 1.0                                                       DFAS 421
      DO 760 J = 2,K                                                 DFAS 422
        R1 = R1*RACUM                                                DFAS 423
        DO 760 I = 1,NY                                              DFAS 424
 760      Y(J,I) = SAVE(J,I)*R1                                       DFAS 425
      DO 770 I = 1,NY                                                DFAS 426
 770    Y(1,I) = SAVE(1,I)                                           DFAS 427


      CALL COPYZ (YL,YLSV,LCOPYL)                                    DFAS 428
      IWEVAL = 1                                                     DFAS 429
      GO TO 250                                                      DFAS 430
C*****************************************************             DFAS 431
C*    THIS SECTION SCALES THE Y DERIVATIVES BY R.                    DFAS 432
C*****************************************************             DFAS 433
 800  R1 = 1.0                                                       DFAS 434
      DO 810 J = 2,K                                                 DFAS 435
        R1 = R1*R                                                    DFAS 436
        DO 810 I = 1,NY                                              DFAS 437
 810      Y(J,I) = Y(J,I)*R1                                          DFAS 438
      GO TO IRET, (100,700)                                          DFAS 439
```

```
C*************************************************                    DFAS 440
C*       ENTER HERE TO RESET VALUES IN PREP FOR AN AUTO-RESTART.     DFAS 441
C*************************************************                    DFAS 442
         ENTRY RESUB                                                 DFAS 443
C        IF INTERRUPT OCCURRED IN MATIN1, NO RESTORATION NECESSARY:  DFAS 444
         IF (KZILCH .EQ. 0) RETURN                                   DFAS 445
         DO 910 J = 1,NY                                             DFAS 446
           Y(1,J) = SAVE(1,J)                                        DFAS 447
 910       Y(2,J) = SAVE(2,J)/HOLD                                   DFAS 448
         CALL COPYZ (YL,YLSV,LCOPYL)                                 DFAS 449
         T(1) = TOLD                                                 DFAS 450
         RETURN                                                      DFAS 451
       END                                                           DFAS 452
```

BLIOGRAPHIC DATA IEET	1. Report No. UIUCDCS-R-73-575	2.	3. Recipient's Accession No.
Title and Subtitle: DOCUMENTATION FOR DFASUB--A Program for the Solution of Simultaneous Implicit Differential and Nonlinear Equations			5. Report Date: July 1973 6.
Author(s): R. L. Brown, C. W. Gear			8. Performing Organization Rept. No.
Performing Organization Name and Address: Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. US AEC AT(11-1)1469
. Sponsoring Organization Name and Address: US AEC Chicago Operations Office, 9800 South Cass Avenue, Argonne, Illinois			13. Type of Report & Period Covered: Technical 14.

. Supplementary Notes

. Abstracts

This report surveys the theory of an efficient method for solving $\underline{f}(\underline{y}, \underline{y}', t) + P(t)\underline{v} = 0$ to get approximations to $\underline{y}(t)$ and $\underline{v}(t)$ at discrete time points $t_i > t_0$ given $\underline{y}(t_0)$ and $\underline{y}'(t_0)$. The organization and use of a program, DFASUB, which uses this theory is described.

. Key Words and Document Analysis. 17a. Descriptors

ordinary differential equations
nonlinear equations
linear equations
discrete variable method
simultaneous implicit differential and nonlinear equations

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

. Availability Statement	19. Security Class (This Report)	21. No. of Pages
unlimited	UNCLASSIFIED	42
	20. Security Class (This Page) UNCLASSIFIED	22. Price

orm AEC-427
(6/68)
AECM 3201

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

. AEC REPORT NO.	2. TITLE
COO-1469-0225	DOCUMENTATION FOR DFASUB--A Program for the Solution of Simultaneous Implicit Differential and Nonlinear Equations

3. TYPE OF DOCUMENT (Check one):

- [X] a. Scientific and technical report
- [ ] b. Conference paper not to be published in a journal:
  - Title of conference _____
  - Date of conference _____
  - Exact location of conference _____
  - Sponsoring organization _____
- [ ] c. Other (Specify) _____

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- [X] a. AEC's normal announcement and distribution procedures may be followed.
- [ ] b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- [ ] c. Make no announcement or distrubution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear
Professor and Principal Investigator

Organization
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Signature	Date
Charles W Gear.	July 1973

FOR AEC USE ONLY

AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

PATENT CLEARANCE:

- [ ] a. AEC patent clearance has been granted by responsible AEC patent group.
- [ ] b. Report has been sent to responsible AEC patent group for clearance.
- [ ] c. Patent clearance not required.